



BeagleBone con Arch Linux

Manual de instalación y manejo

Ignacio Alvarez García – Septiembre 2014

INDICE





1. Instalación del Sistema Operativo Arch Linux

(de: http://archlinuxarm.org/platforms/armv7/beaglebone)

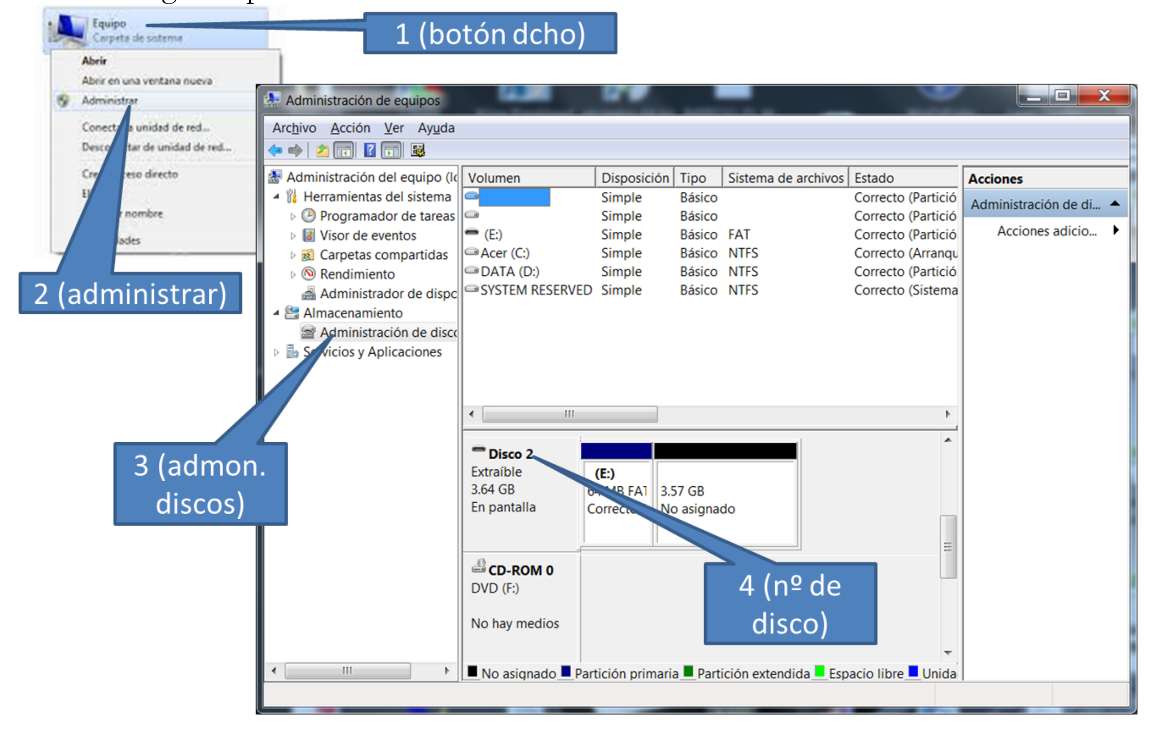
Estos pasos ya están realizados en la tarjeta SD disponible.

Fuentes consultadas:
http://qt-project.org/wiki/RaspberryPi_Beginners_guide
http://framboisepi.fr/installation-archlinux/
http://elinux.org/ArchLinux_Install_Guide
http://www.andremiller.net/content/mounting-hard-disk-image-including-partitions-using-linux
http://www.ics.com/blog/building-qt-5-raspberry-pi
http://florisdriessen.nl/electronics/getting-qt5-with-opengl-es-20-support-on-raspberry-pi/
http://ebalaskas.gr/blog/2013/03/25/raspberry-pi-with-archlinux-under-qemu/

1.1. Crear tarjeta SD con ArchLinux para BeagleBone

1.1.1. Desde host Windows

1) Introducir tarjeta Micro-SD en el ordenador host, y comprobar el nº de disco que ha sido asignado por Windows mediante el administrador de discos:



2) Ejecutar ventana de comandos Windows como administrador (menú Windows, Símbolo del Sistema, botón derecho, ejecutar como administrador), y en la misma escribir el comando siguiente:

```
C:\Windows\system32> c:\Program Files\Oracle\VirtualBox\VBoxManage.exe internalcommands createrawvmdk -filename C:\XXX\sd-card.vmdk -rawdisk \\.\PhysicalDriveN
RAW host disk access VMDK file C:\XXX\sd-card.vmdk created successfully.
```

Donde XXX debe sustituirse por un directorio local deseado, y N por el nº de disco (2 en este caso).

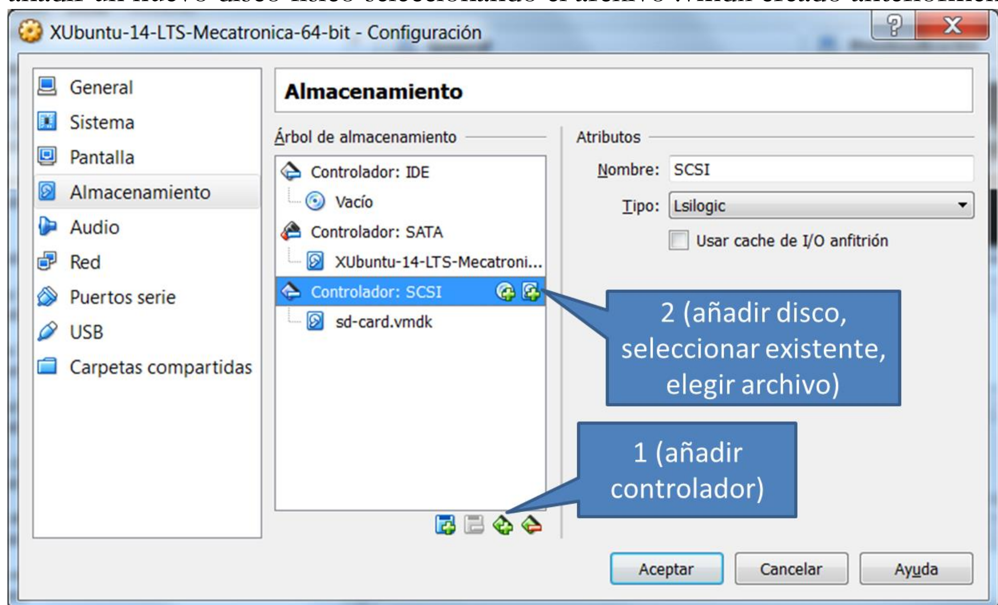
3) Desde la misma ventana de comandos Windows como administrador, eliminar todas las particiones de la tarjeta SD:

```
C:\Windows\system32> diskpart
DISKPART> list
( Comprobar el nº de disco N )
DISKPART> select disk N
DISKPART> clean
DISKPART> exit
```

Donde N por el nº de disco (2 en este caso).

1.1.2. Desde configuración de VirtualBox

4) Ejecutar VirtualBox como administrador (botón derecho, ejecutar como administrador).
5) Seleccionar la máquina virtual, a continuación Configuración … Almacenamiento, y añadir un nuevo disco físico seleccionando el archivo .vmdk creado anteriormente.



6) Aceptar y lanzar máquina virtual.

1.1.3. Desde guest Linux VirtualBox

7) Ejecutar terminal de comandos, y en la misma comprobar el dispositivo en el que se ha alojado el disco SD (típicamente será /dev/sdb, pero lo indicaremos a partir de ahora de forma genérica con /dev/sdX) mediante el comando:

```
$ sudo lshw -class disk        (pedirá clave)
  *-disk
     description: SCSI Disk
       …
     logical name: /dev/sdX
     size: 3724MiB (3904MB)
       …
  *-cdrom
       …
  *-disk
     description: ATA Disk
```

```
 …
logical name: /dev/sda
…
```

8) Preparar directorio para descargas e instalaciones, y crear variable de entorno para indicarlo, ej:

```
$ export BASE_DIR = /home/developer/proyectos/Qt5OnBB
$ mkdir $BASE_DIR
$ cd $BASE_DIR
$ mkdir Installs
$ mkdir Downloads
$ mkdir –p Mounts/RootFS
$ mkdir –p Mounts/Boot
```

9) Descargar ArchLinux (bootloader y root filesystem) para Beaglebone desde `http://archlinuxarm.org/platforms/armv7/ti/beaglebone` sobre el directorio `$BASE_DIR/Downloads`. Formatear el disco SD y copiar archivos siguiendo las instrucciones de dicha página como superusuario (sustituir la X por la letra que se ha obtenido anteriormente):

```
$ cd $BASE_DIR/Downloads
$ sudo su        (pedirá clave)
#        (el prompt # indica que se está trabajando como superusuario)
# fdisk /dev/sdX
Command: o ↵
Command: p ↵
Command: n ↵
Select: p ↵
Partition number: 1 ↵
First sector: ↵
Last sector: +64M  ↵
Command: t ↵
Hex code: e ↵
Command : a ↵
Partition number: 1 ↵
Command : n ↵
Select : p ↵
Partition number:  2 ↵
First sector: ↵
Last sector:   ↵
w ↵
# umount /dev/sdX1
# umount /dev/sdX2
# mkfs.vfat  -F  16  /dev/sdX1
# mkfs.ext4  /dev/sdX2
# mkdir boot
```

```
# tar  -xvf  BeagleBone-bootloader.tar.gz  -C  boot
# mount  /dev/sdX1  ../Mounts/Boot
# cp boot/MLO  ../Mounts/Boot
# cp boot/u*  ../Mounts/Boot
# umount  .. /Mounts/Boot
# mount  /dev/sdX2  .. /Mounts/RootFS
# tar  -xvf  ArchLinuxARM-am33x-latest.tar.gz  -C  ../Mounts/RootFS
```

10) Editar archivo para configuración estática de red :

```
# mousepad ../Mounts/RootFS/etc/systemd/network/eth0.network
 [Match]
Name=eth0

[Network]
Address=192.168.100.1/24
Gateway=192.168.100.255
# umount ../Mounts/RootFS
# tar  -xvf  ArchLinuxARM-am33x-latest.tar.gz  -C  ../Mounts/RootFS
```

11) Apagar la máquina virtual, y extraer desde el host de forma segura la tarjeta SD.

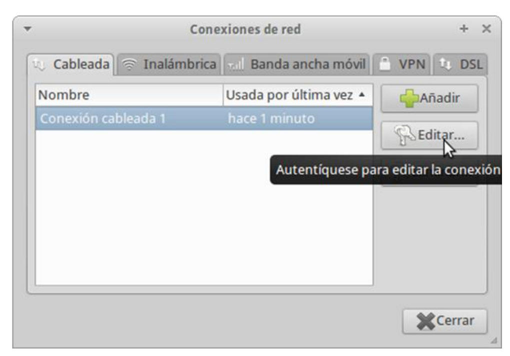
12) Introducir la tarjeta SD creada y arrancar Beaglebone. Al cabo de unos segundos, deberían verse los LED de la tarjeta parpadeando. Si no es así, ha habido algún problema en la instalación: repetir los pasos anteriores desde un Linux no virtual (en este caso el disco aparecerá típicamente como /dev/mmcblk0, y las particiones como /dev/mmcblk0p1 y /dev/mmcblk0p2).

13) Eliminar en la configuración del guest VirtualBox la conexión creada a la tarjeta SD en el paso 5.

14) Preparar en el guest VirtualBox una 2ª conexión de red en la misma subred que la BeagleBone (añadir adaptador de la misma forma que se ha realizado con el primero, y editar sus propiedades para poner una dirección fija 192.168.100.50):

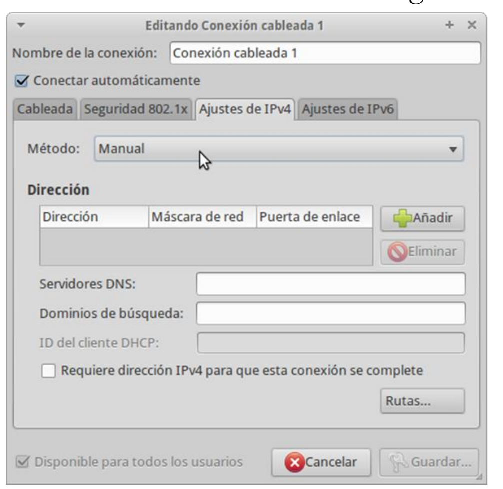
   a. Asegurar que se utiliza conexión de red con adaptador puente en la configuración de VirtualBox (apartado 1.4 en documento de instalación de Linux bajo VirtualBox).
   b. Editar la conexión de red cableada pulsando el icono de red en la parte superior derecha de la consola, seleccionando la conexión de red y pulsando Editar.

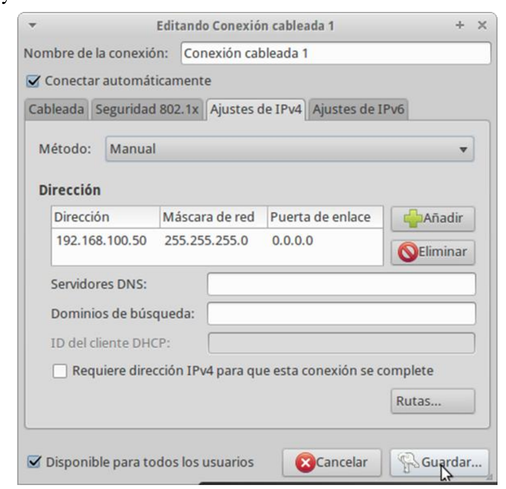




c. En las propiedades de la conexión de red, patilla "Ajustes de IPV4", seleccionar Manual y pulsar "Añadir dirección".

d. Añadir una dirección en el rango 192.168.100.xx (excepto la 1), con máscara de red 255.255.255.0. Pulsar guardar y cerrar.





15) Desde terminal de comandos, acceder a la BB mediante ssh, y modificar la clave de root:

```
$ ssh root@192.168.1.1
  password: root
[root@alarm ~]# passwd
Enter new UNIX password: unioviisa2014
[root@alarm ~]# exit
$
```

16) Descargar tools-master.zip de https://github.com/raspberrypi/tools [Download ZIP] en directorio Downloads, y descomprimir en directorio Installs:

```
$ export BASE_DIR=/home/developer/proyectos/Qt5OnBB
$ cd $BASE_DIR/Installs
$ unzip ../Downloads/tools-master.zip
```

17) Instalar git y descargar qt5:

```
$ sudo apt-get install git
$ cd $BASE_DIR/Downloads
$ mkdir -p git
```

```
$ cd git
$ git clone git://gitorious.org/qt/qt5.git
$ cd qt5
$ ./init-repository
```

18) Compilación cruzada de qt5 para BeagleBone. Descargar fixQualifiedLibraryPaths http://gitorious.org/cross-compile-tools/cross-compile-tools/ en $BASE_DIR_Installs/cross-compile-tools. A continuación, compilar:

```
$ cd $BASE_DIR/Installs/cross-compile-tools
$ ./fixQualifiedLibraryPaths $BASE_DIR/Mounts/RootFS $BASE_DIR/Installs/tools-master/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf-gcc
$ cd $BASE_DIR/Downloads/git/qt5
$ ./configure -v -device linux-rasp-pi-g++ -device-option
CROSS_COMPILE=$BASE_DIR/Installs/tools-master/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf-
-sysroot $BASE_DIR/Mounts/RootFS -opensource -confirm-license
-optimized-qmake -reduce-exports -release -make libs -nomake examples
-qt-zlib -qt-libpng -qt-libjpeg -prefix /opt/Qt530/5.3/BB
$  make
$ sudo make install
((( En caso de querer rehacer la instalación con otra configuración, borrar la configuración previa con: $ git submodule foreach --recursive "git clean -dfx" )))
```

19) Instalación de depurador remoto. Descargar fuentes de gdb de http://ftp.gul.es/gnu/ftp.gnu.org/gnu/gdb/ en directorio $BASE_DIR/Downloads. A continuación:

```
$ sudo apt-get install gdb-multiarch
$ cd $BASE_DIR/Installs
$ tar –xvf $BASE_DIR/Downloads/gdb-x.x.tar.gz
$ cd gdb-x.x/gdb/gdbserver
$ ./configure --host=arm-linux-gnueabihf
--prefix=$BASE_DIR/Mounts/RootFS/usr
CC=$BASE_DIR/Installs/tools-master/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf-gcc
$ make
$ sudo make install
```

20) Copia de los archivos instalados a la tarjeta SD mediante el administrador de archivos:

Ventana origen: $BASE_DIR/Mounts/RootFS/ (sustituir $BASE_DIR por su contenido)
Ventana destino: sftp://root@192.168.100.1/

Copiar: opt/Qt530 (origen) a /opt (destino)



Copiar: usr/bin/gdbserver (origen) a /usr/bin (destino)

1.1.4. Pruebas para comprobar que la instalación ha sido correcta

Descargar:
http://isa.uniovi.es/~ialvarez/Curso/Mecatronica/C3-FabricacionSistemasMecatronicos/descargas/TestsQt5OnBB.tar.gz
y descomprimir en directorio $BASE_DIR.

21) Pruebas cross compile:

18.a) gcc modo consola

```
$ cd $BASE_DIR/Tests/Console_gcc
$ make
Copiar ejecutables helloworld_cpp y helloworld_c a la tarjeta SD
(directorio /root)
Iniciar sesión remota en BB con:
$ ssh root@192.168.100.1
Ejecutar:
# ./helloworld_cpp
# ./helloworld_c
# exit
```

18.b) gcc modo consola con QtCore

```
$ cd $BASE_DIR/Tests/Console_qt
$ make
Copiar ejecutable helloworld_qt a la tarjeta SD (directorio /root)
Iniciar sesión remota en BB con:
$ ssh root@192.168.100.1
Ejecutar:
# export LD_LIBRARY_PATH=/opt/Qt530/5.3/BB/lib
# ./helloworld_qt
# exit
```

18.c) Prueba debug remoto

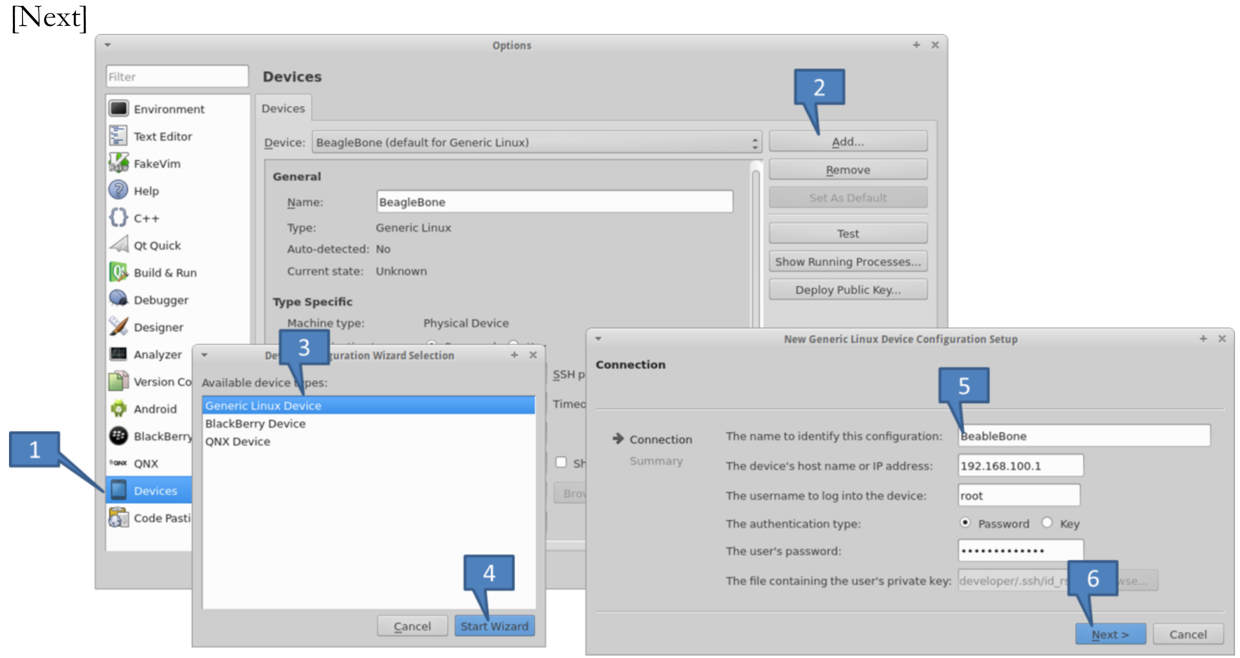
```
$ cd $BASE_DIR/Tests/DbgServer
$ make
Copiar ejecutable dbg_test_c a la tarjeta SD (directorio /root)
Iniciar sesión remota en BB (desde nueva consola) con:
$ ssh root@192.168.100.1
Ejecutar:
# gdbserver 192.168.100.1:22552 dbg_test_c
En la sesión local:
$ gdb-multiarch -x gdb_cmds.txt dbg_test_c
(gdb) break main
(gdb) continue
(gdb) step... (ver resultados y escribir entradas en consola BB)
(gdb) print x (o y)
(gdb) quit
```



2. Configurar kit de compilación cruzada en QtCreator

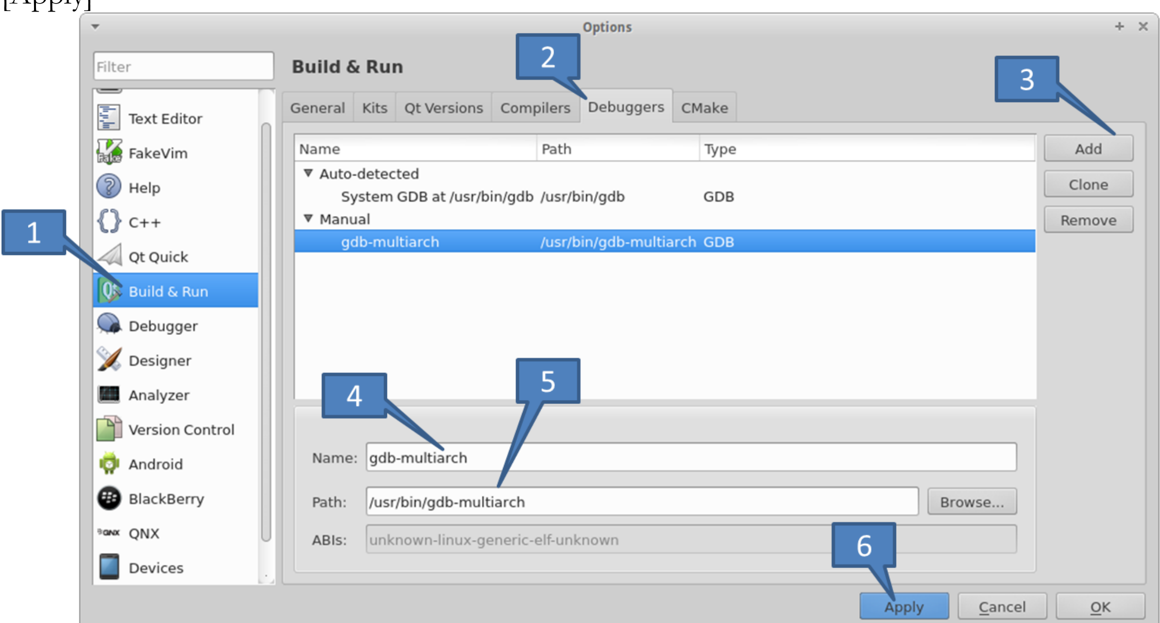
2.1. Crear nuevo kit

Arrancar QtCreator, e ir a Tools -> Options
* Devices -> Add... Generic Linux Device
Name : BeagleBone
IP : 192.168.100.1
User : root
Pass: unioviisa2014
[Next]

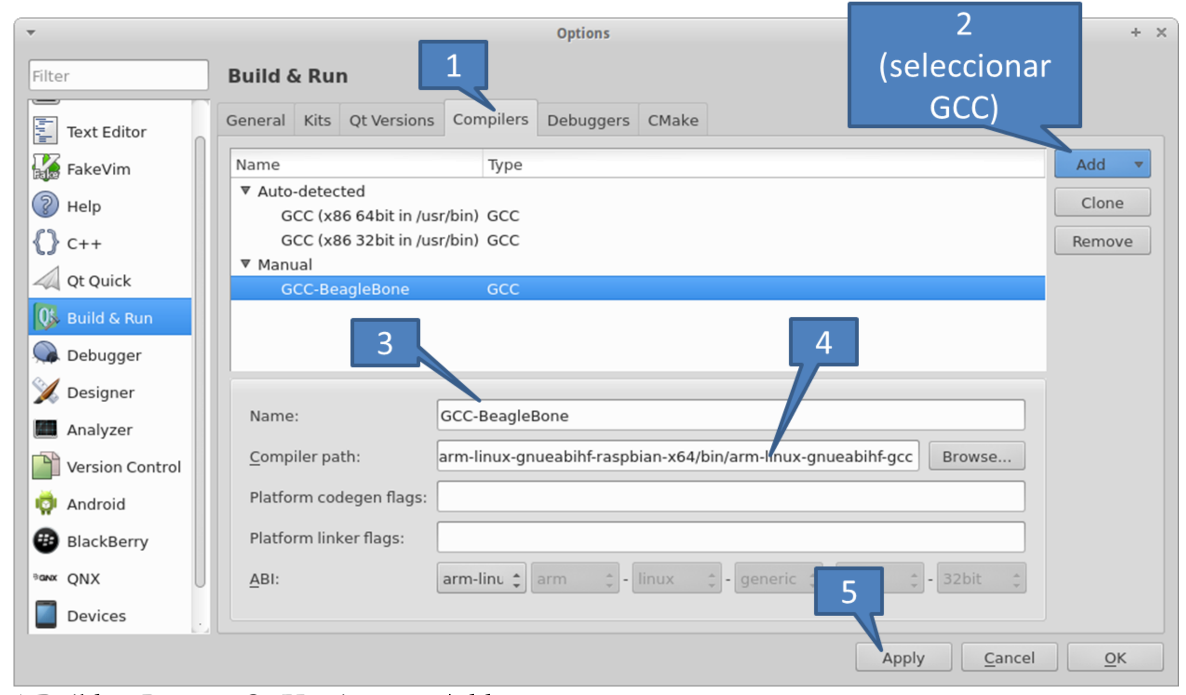


Tras comprobaciones...
GDB server executable: gdbserver
Apply

* Build & Run -> Debuggers -> Add...
Name: gdb-BeagleBone
Path: gdb-multiarch
[Apply]

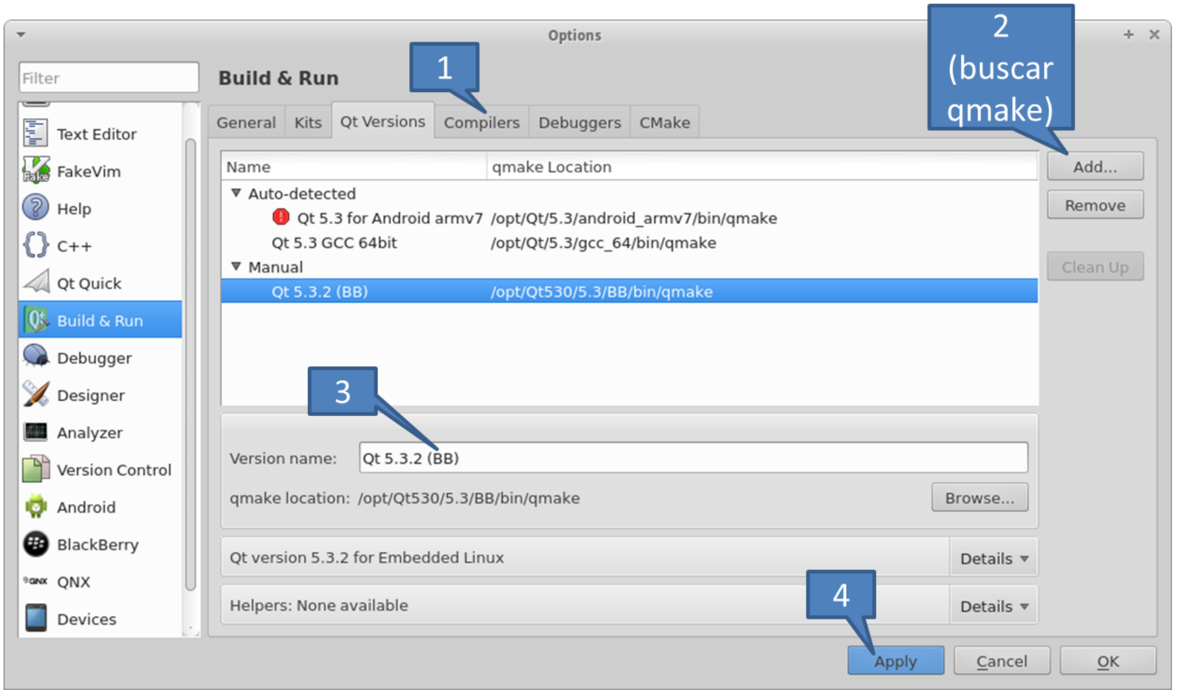


* Build & Run -> Compilers -> Add...GCC
Name: gcc-BeagleBone
Path:
/home/developer/proyectos/Raspberry-Qt5-ArchLinux/Installs/tools-master/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf-gcc
(modificar según donde se encuentre el directorio Installs)
[Apply]

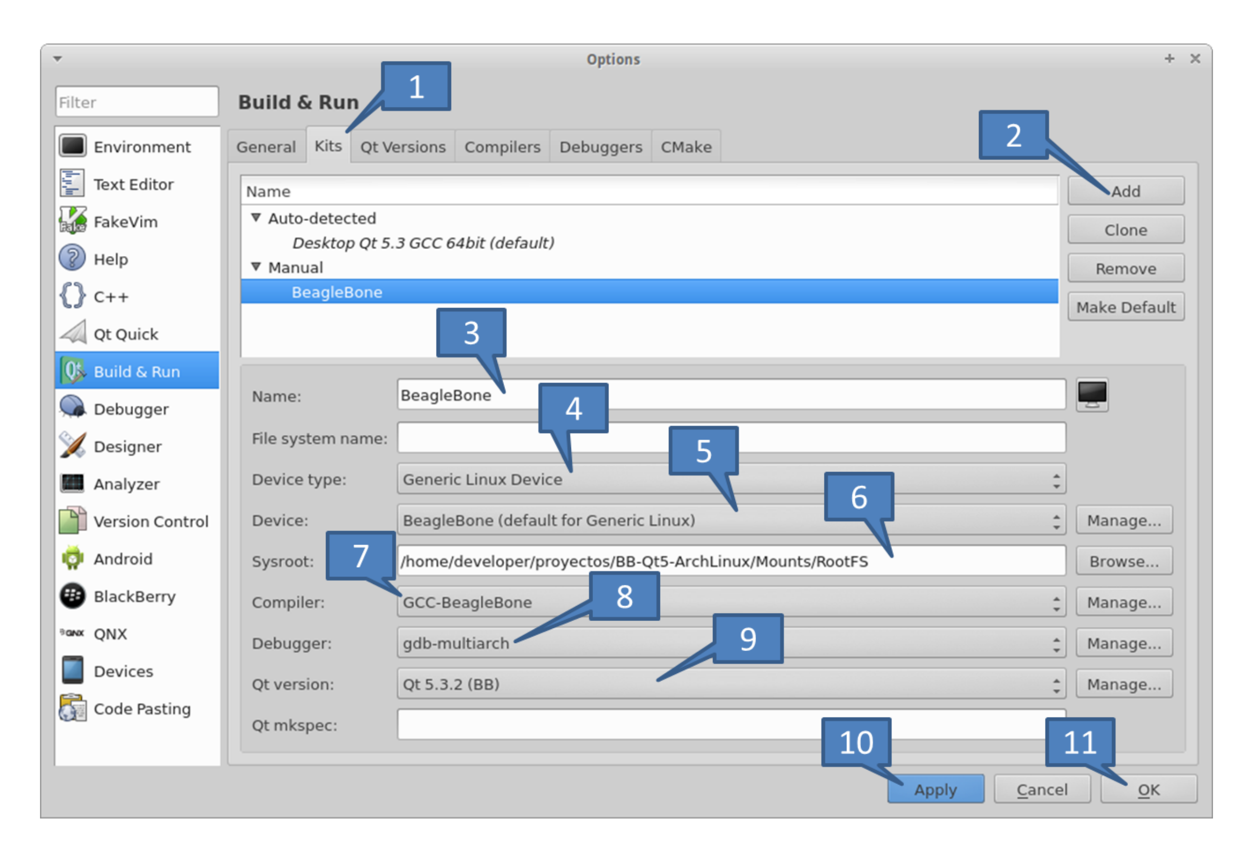


* Build & Run -> Qt Versions -> Add...
En ventana de archivos, buscar qmake: /opt/Qt530/5.3/BB/bin/qmake

Dar nombre Qt 5.3.2 (BB)
[Apply]



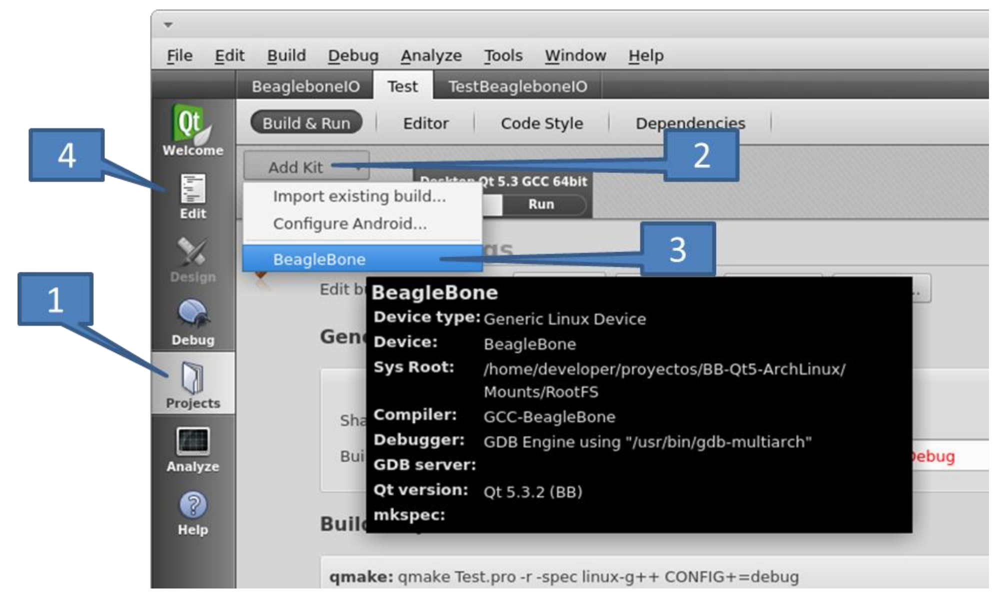
* Build & Run -> Kits -> Add...
Name: BeagleBone
Device type: Generic Linux Device
Device: BeagleBone
Sysroot: /home/developer/proyectos/BB-Qt5-ArchLinux/Mounts/RootFS (cambiar en función de localización del directorio)
Compiler: gcc-BeagleBone
Debugger: gdb-BeagleBone
Qt version: Qt 5.3.2 (BB)
[Apply] y [Ok] para terminar



2.2. Aplicar kit al proyecto

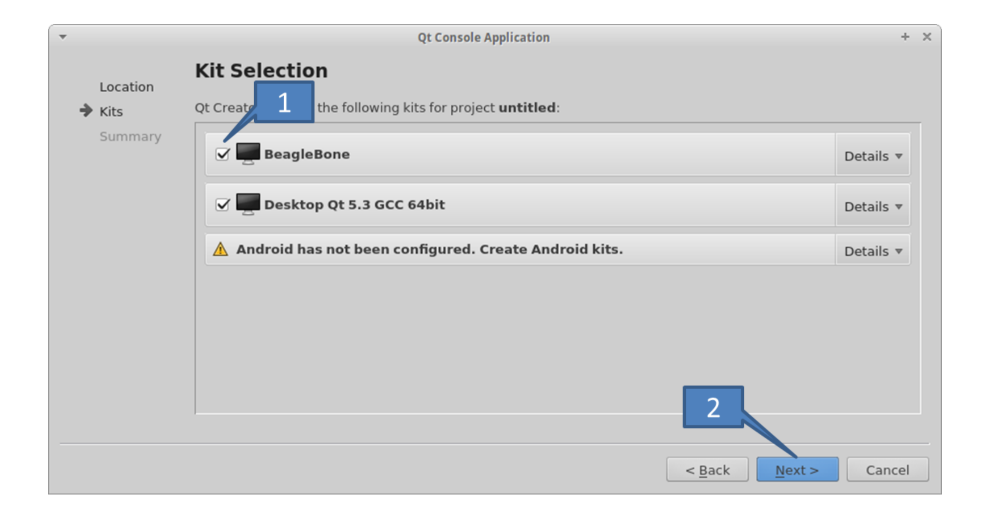
2.2.1. Añadir a proyecto existente

Seleccionar Projects en los iconos a la izquierda, a continuación Add Kit y seleccionar Beablebone. A continuación, volver a Edit.



2.2.2. Crear nuevo proyecto

Al crear nuevo proyecto, aparecerá una pantalla de selección de Kit, elegir Beaglebone y continuar.



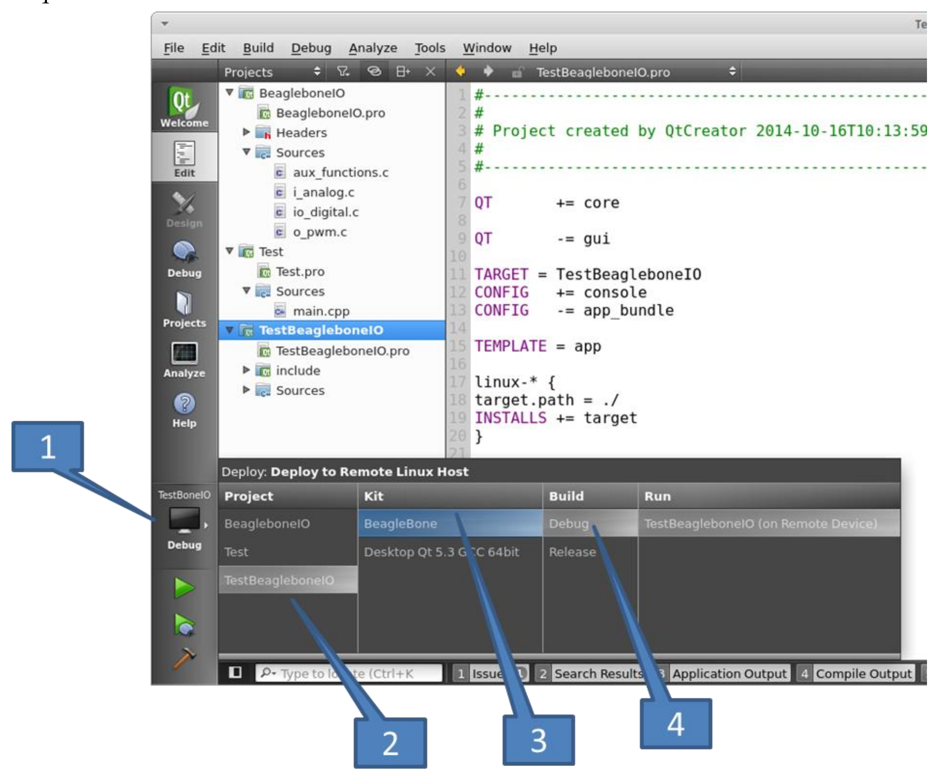
2.2.3. Modificar archivo .pro (ambos casos)

Seleccionar archivo .pro del proyecto, y añadir las líneas siguientes:

```
linux-* {
target.path = ./
INSTALLS += target
}
```

2.2.4. Seleccionar kit de compilación a utilizar (ambos casos)

Seleccionar el kit deseado para compilación y ejecución en el icono de selección situado a la izquierda del interface:



La compilación y ejecución se realizará en el equipo remoto, y se puede ejecutar también en modo depuración.

2.3. Instalar la aplicación para que se ejecute en el arranque

La próxima vez que se reinicie el BeagleBone se ejecutará el programa directamente.

3. Acceso a dispositivos de E/S

Fuentes consultadas (atención a las diferencias entre Beaglebone y Beaglebone-Black):
http://www.adminempire.com/beaglebone-basics-for-arch-linux/
http://www.armhf.com/using-beaglebone-black-gpios/
http://derekmolloy.ie/gpios-on-the-beaglebone-black-using-device-tree-overlays/

Los dispositivos de E/S habituales de los sistemas Linux (consola, puerto serie, conexión Ethernet, etc.) se acceden en BeagleBone de igual manera que en otros equipos con el mismo S.O.

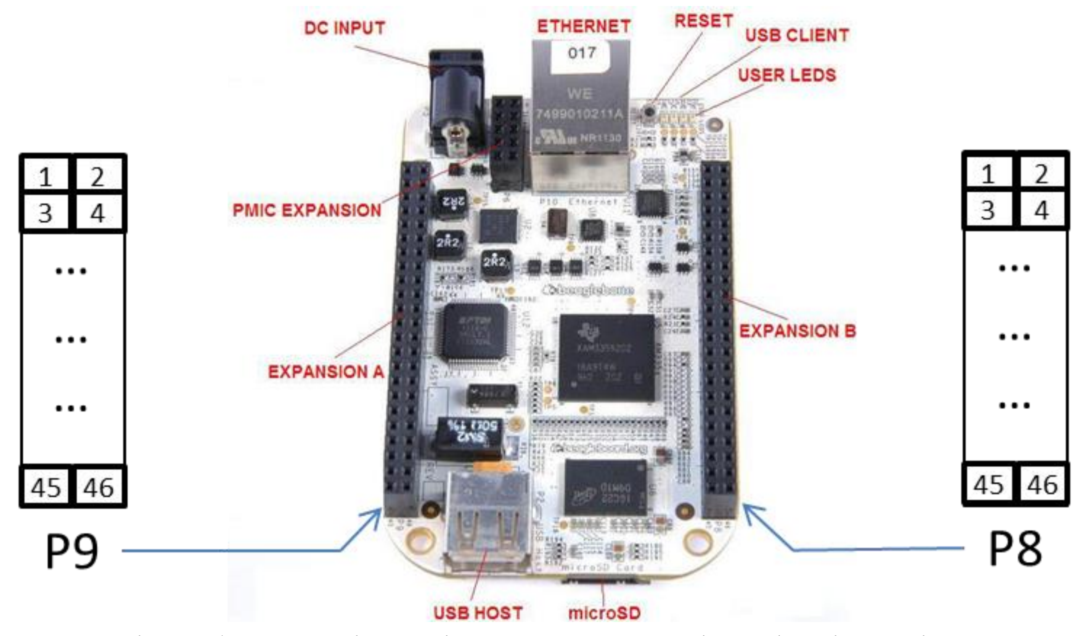
Sin embargo, los dispositivos de E/S más específicos, agrupados bajo el nombre genérico de GPIO – General Purpose Input Output - (E/S digital, entrada analógica, salida PWM) requieren un trabajo más laborioso, que conlleva la lectura de documentación dispersa y en muchos casos sólo existente a partir de respuestas en foros.

A continuación se resumen los requerimientos y procedimientos para esta E/S, y se acompaña una librería de funciones que facilitan su manejo (ver sección 3.3).

3.1. Interfaz hardware

3.1.1. Conectores de expansión P8 y P9

Existen 2 conjuntos de pines de expansión en los conectores P8 y P9, documentados en el manual de referencia (ver 3.1.3). En cada uno de los pines de estos conectores se exponen múltiples señales de la CPU ARM, por lo que existe un multiplexor que permite seleccionar cuál de las señales es accesible en cada momento.



Las señales disponibles en cada pin de estos conectores dependen del modo seleccionado en el multiplexor, según la tabla siguiente:

CONECTOR P8

PIN	MODE							
	0	1	2	3	4	5	6	7
P8_1	GND							
P8_2	GND							
P8_3	gpmc_ad6	mmc1_dat6						GPIO1_6
P8_4	gpmc_ad7	mmc1_dat7						GPIO1_7
P8_5	gpmc_ad2	mmc1_dat2						GPIO1_2
P8_6	gpmc_ad3	mmc1_dat3						GPIO1_3
P8_7	gmpc_advn_ale		timer4					GPIO2_2
P8_8	gpmc_oen_ren		timer7					GPIO2_3
P8_9	gpmc_be0n_cle		timer5					GPIO2_5
P8_10	gpmc_wen		timer6					GPIO2_4
P8_11	gpmc_ad13	lcd_data18	mmc1_dat5	mmc2_dat1	eQEP2B_in			GPIO1_13
P8_12	gpmc_ad12	lcd_data19	mmc1_dat4	mmc2_dat0	EQEP2A_IN			GPIO1_12
P8_13	gpmc_ad9	lcd_data22	mmc1_dat1	mmc2_dat5	ehrpwm2B			GPIO0_23
P8_14	gpmc_ad10	lcd_data21	mmc1_dat2	mmc2_dat6	ehrpwm2_trip zone_in			GPIO0_26
P8_15	gpmc_ad15	lcd_data16	mmc1_dat7	mmc2_dat3	eQEP2_strobe			GPIO1_15
P8_16	gpmc_ad14	lcd_data17	mmc1_dat6	mmc2_dat2	eQEP2_index			GPIO1_14
P8_17	gpmc_ad11	lcd_data20	mmc1_dat3	mmc2_dat7	ehrpwm0_synco			GPIO0_27
P8_18	gpmk_clk_mux0	lcd_memory_clk	gpmc_wait1	mmc2_clk			mcasp0_fsr	GPIO2_1
P8_19	gpmc_ad18	lcd_data23	mmc1_dat0	mmc2_dat4	ehrpwm2A			GPIO0_22
P8_20	gpmc_csn2	gpmc_be1n	mmc1_cmd					GPIO1_31
P8_21	gpmc_csn1	gpmc_clk	mmc1_clk					GPIO1_30
P8_22	gpmc_ad5	mmc1_dat3						GPIO1_5
P8_23	gpmc_ad4	mmc1_dat4						GPIO1_4
P8_24	gpmc_ad1	mmc1_dat1						GPIO1_1
P8_25	gpmc_ad0	mmc1_dat0						GPIO1_0
P8_26	gpmc_csn0							GPIO1_29
P8_27	lcd_vsync	gpmc_a8						GPIO2_22
P8_28	lcd_pclk	gpmc_a10						GPIO2_24
P8_29	lcd_hsync	gpmc_a9						GPIO2_23
P8_30	lcd_ac_bias_en	gpmc_a11						GPIO2_25
P8_31	lcd_data14	gpmc_a18	eQEP1_index	mcasp0_axr1	uart5_rxd		uart5_ctsn	GPIO0_10
P8_32	lcd_data15	gpmc_a19	eQEP1_strobe	mcasp0_ahclkx	mcasp0_axr3		uart5_rtsn	GPIO0_11
P8_33	lcd_data13	gpmc_a17	eQEP1B_in	mcasp0_fsr	mcasp0_axr3		uart4_rtsn	GPIO0_9
P8_34	lcd_data11	gpmc_a15	ehrpwm1B	mcasp0_ahclkr	mcasp0_axr2		uart3_rtsn	GPIO2_17
P8_35	lcd_data12	gpmc_a16	eQEP1A_in	mcasp0_aclkr	mcasp0_axr2		uart4_ctsn	GPIO0_8
P8_36	lcd_data10	gpmc_a14	ehrpwm1A	mcasp0_axr0			uart3_ctsn	GPIO2_16
P8_37	lcd_data8	gpmc_a12	ehrpwm1_trip zone_in	mcasp0_aclkx	uart5_txd		uart2_ctsn	GPIO2_14
P8_38	lcd_data9	gpmc_a13	ehrpwm0_synco	mcasp0_fsx	uart5_rxd		uart2_rtsn	GPIO2_15
P8_39	lcd_data6	gpmc_a6		eQEP2_index				GPIO2_12
P8_40	lcd_data7	gpmc_a7		eQEP2_strobe	pr1_edio_data_out7			GPIO2_13
P8_41	lcd_data4	gpmc_a4		eQEP2A_in				GPIO2_10
P8_42	lcd_data5	gpmc_a5		eQEP2B_in				GPIO2_11
P8_43	lcd_data2	gpmc_a2		ehrpwm2_trip zone_in				GPIO2_8
P8_44	lcd_data3	gpmc_a3		ehrpwm0_synco				GPIO2_9
P8_45	lcd_data0	gpmc_a0		ehrpwm2A				GPIO2_6
P8_46	lcd_data1	gpmc_a1		ehrpwm2B				GPIO2_7

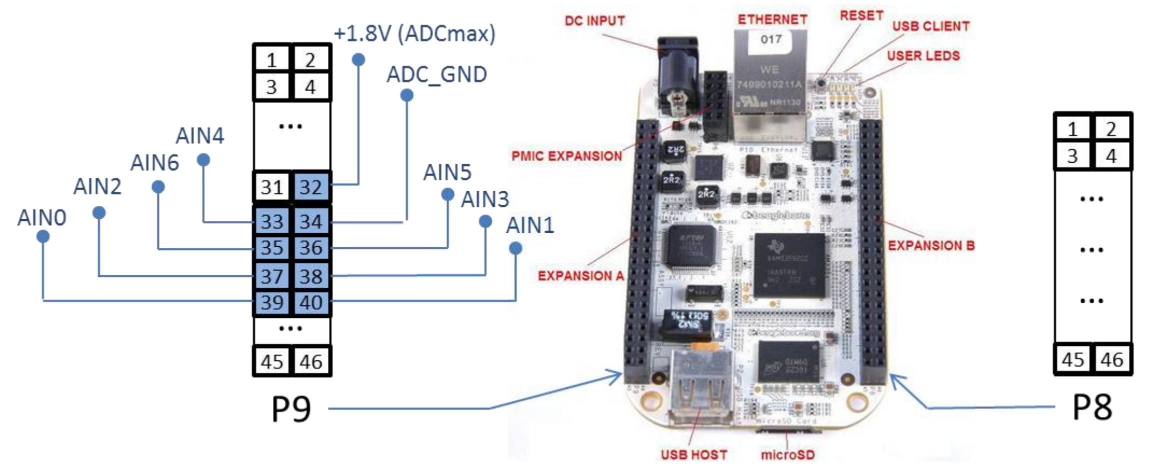
CONECTOR P9

PIN	MODE							
	0	1	2	3	4	5	6	7
P9_1	GND							
P9_2	GND							
P9_3	VDD_3V3EXP							
P9_4	VDD_3V3EXP							
P9_5	VDD_5V							
P9_6	VDD_5V							
P9_7	SYS_5V							
P9_8	SYS_5V							
P9_9	PWR_BUT							
P9_10	RESET_OUT							
P9_11	gpmc_wait0	mii2_crs	gpmc_csn4	rmii2_crs_dv	mmc1_sdcd		uart4_rxd_mux2	GPIO0_30
P9_12	gpmc_be1n	mii2_col	gpmc_csn6	mmc2_dat3	gpmc_dir		mcasp0_aclkr_mux3	GPIO1_28
P9_13	gpmc_wpn	mii2_rxerr	gpmc_csn5	rmii2_rxerr	mmc2_sdcd		uart4_txd_mux2	GPIO0_31
P9_14	gpmc_a2	mii2_txd3	rgmii2_td3	mmc2_dat1	gpmc1_a18		ehrpwm1A_mx1	GPIO1_18
P9_15	gpmc_a0	gmii2_txen	rmii2_tctl	mii2_txen	gpmc_a16		ehrpwm1_tripzone_input	GPIO1_16
P9_16	gpmc_a3	mii2_txd2	rgmii2_td2	mmc2_dat2	gpmc_a19		ehrpwm1B_mx1	GPIO1_19
P9_17	spi0_cs0	mmc2_sdwp	I2C1_SCL	ehrpwm0_synci				GPIO0_5
P9_18	spi0_d1	mmc1_sdwp	I2C1_SDA	ehrpwm0_tripzone				GPIO0_4
P9_19	uart1_rtsn	timer5	dcan0_rx	I2C2_SCL	spi1_cs1			GPIO0_13
P9_20	uart1_ctsn	timer6	dcan0_tx	I2C2_SDA	spi1_cs0			GPIO0_12
P9_21	spi0_d0	uart2_txd	I2C2_SCL	ehrpwm0B			EMU3_mux1	GPIO0_3
P9_22	spi0_sclk	uart2_rxd	I2C2_SDA	ehrpwm0A			EMU2_mux1	GPIO0_2
P9_23	gpmc_a1	gmii2_rxdv	rgmii2_rxdv	mmc2_dat0	gpmc_a17		ehrpwm0_synco	GPIO1_17
P9_24	uart1_txd	mmc2_sdwp	dcan1_rx	I2C1_SCL				GPIO0_15
P9_25	mcasp0_ahclkx	eQEP0_strobe	mcasp0_axr3	mcasp1_axr1	EMU4_mux2			GPIO3_21
P9_26	uart1_rxd	mmc1_sdwp	dcan1_tx	I2C1_SDA				GPIO0_14
P9_27	mcasp0_fsr	eQEP0B_in	mcasp0_axr3	mcasp1_fsx	EMU2_mux2			GPIO3_19
P9_28	mcasp0_ahclkr	ehrpwm0_synci	mcasp0_axr2	spi1_cs0	eCAP2_in_PWM2_out			GPIO3_17
P9_29	mcasp0_fsx	ehrpwm0B		spi1_d0	mmc1_sdcd_mux1			GPIO3_15
P9_30	mcasp0_axr0	ehrpwm0_tripzon		spi1_d1	mmc2_sdcd_mux1			GPIO3_16
P9_31	mcasp0_aclkx	ehrpwm0A		spi1_sclk	mmc0_sdcd_mux1			GPIO3_14
P9_32	VDD_ADC_1.8V							
P9_33	AIN4							
P9_34	ADC_GND							
P9_35	AIN6							
P9_36	AIN5							
P9_37	AIN2							
P9_38	AIN3							
P9_39	AIN0							
P9_40	AIN1							
P9_41	xdma_event_intr1		tclkin	clkout2	timer7_mux1		EMU3_mux0	GPIO0_20
P9_42	eCAP0_in_PWM0_out	uart3_txd	spi1_cs1	pr1_ecap0_ecap_capin_apwm_o	spi1_sclk	mmc0_sdwp	xdma_event_intr2	GPIO0_7
P9_43	GND							
P9_44	GND							
P9_45	GND							
P9_46	GND							

3.1.2. Entradas analógicas

Existen 7 entradas analógicas están disponibles en los pines AINxx en el conector P9, con 12 bits de resolución (valores 0 a 4095), y frecuencia de muestreo máxima de 100Ks/s (aunque ésta se puede reducir debido al driver de acceso).

¡¡¡ OJO!!! Estas entradas tienen un rango de tensión de **0V a 1.8V**; se daña la placa si se utilizan con una tensión mayor.



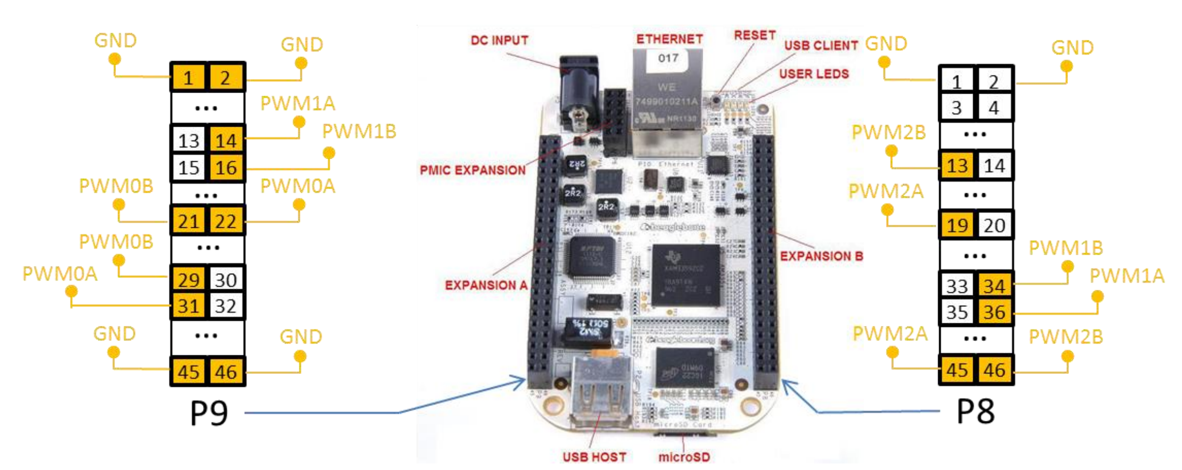
Las entradas analógicas se leen mediante las funciones siguientes de la librería suministrada (ver apartado 3.3):

- int InitAnalogInput(enumAIpin pin);
- int ReadAnalogInput(int pinId);
- int EndAnalogInput(int pinId);

3.1.3. Salidas PWM

Existen 3 salidas PWM, que se activan seleccionando el modo adecuado en los pines correspondientes:

- EHRPWM0A: pin P9_22 (modo 3) ó pin P9_31 (modo 1).
- EHRPWM0B: pin P9_21 (modo 3) ó pin P9_29 (modo 1).
- EHRPWM1A: pin P8_36 (modo 2) ó pin P9_14 (modo 6).
- EHRPWM1B: pin P8_34 (modo 2) ó pin P9_16 (modo 6).
- EHRPWM2A: pin P8_19 (modo 4) ó pin P8_45 (modo 3).
- EHRPWM2B: pin P8_13 (modo 4) ó pin P8_46 (modo 3).



Para que funcionen las salidas PWM, es necesario previamente activar un reloj para el periodo, indicado en ns.

¡¡¡ **OJO!!!** Estos pines funcionan a **3.3V**; se daña la placa si se utilizan con una tensión mayor. La máxima corriente que se debe solicitar es de **6 mA**.

Las salidas PWM se utilizan mediante las funciones siguientes de la librería suministrada (ver apartado 3.3):

- Int InitPWM(int period_ns);
- int InitPWMOutput(enumPWMpin pin);
- int WritePWMOutput(int pwm_id,float duty);
- int EndPWMOutput(int pwm_id);

3.1.4. E/S digitales

Se pueden utilizar múltiples pines para E/S digital a través de los conectores P8 y P9 (todos los indicados con GPIOx_xx en el modo 7. Seleccionar para estos fines aquellos pines que no se piensan utilizar en otros (ej. salida PWM).

¡¡¡ **OJO!!!** Estos pines funcionan a **3.3V**; se daña la placa si se utilizan con una tensión mayor. Cuando están configuradas como salida, la máxima corriente que se debe solicitar es de **6 mA**.

La configuración por defecto de la mayoría de pines es pulldown, esto es, puesto a 0 a través de resistencia.

La activación y configuración de estos pines se realiza mediante las funciones de la librería suministrada (ver apartado 3.3):

- int InitDigitalIO(enumDIOpin pin,enumDIOtype io);
- int ReadDigitalInput(int pinId);
- int WriteDigitalOutput(int pinId,int value);
- int EndDigitalIO(int pinId);

3.2. Acceso a la E/S mediante Linux fs (filesystem)

Esta sección ya está implementada en la librería disponible, sólo es a título informativo.

En todos los casos, los dispositivos aparecen ante el usuario como archivos de texto. En caso de escritura, debe añadirse siempre el retorno de carro ('\n').

Si sólo se desea utilizar la librería suministrada, pasar directamente a la sección 3.3.

3.2.1. Entradas analógicas

Para leer alguna de las entradas analógicas (0 a 7), ejecutar:



echo cape-bone-iio > /sys/devices/bone_capemgr.8/slots

Las entradas analógicas son legibles (12 bits, esto es, valor 0 a 4095) en los archivos de tipo texto:

/sys/bus/iio/devices/iio:device0/in_voltageX_raw

Ejemplo:

cat /sys/bus/iio/devices/iio\:device0/in_voltage0_raw

1656

3.2.2. Manejo de LEDs de placa

Para escribir los LEDs de la placa (0 a 3):

Escribir modo trigger = none en /sys/class/leds/beaglebone\:green\:usrX/trigger

Escribir 0 ó 1 en /sys/class/leds/beaglebone\:green\:usrX/brightness

Ejemplo LED1:

echo none > /sys/class/leds/beaglebone\:green\:usr1/trigger

echo 1 > /sys/class/leds/beaglebone\:green\:usr1/brightness

echo 0 > /sys/class/leds/beaglebone\:green\:usr1/brightness

3.2.1. Manejo de E/S digitales

Los pines GPIO no conectados a otros elementos pueden ser utilizados como entradas o salidas digitales de propósito general. Para utilizarlos, se debe:

- Determinar la relación entre el nº de pin GPIO y los conectores disponibles (P8/P9).
- Calcular el nº absoluto de pin, en función del banco y nº dentro del banco, con la fórmula:

  pinAbs = banco * 32 + pinBanco

  Ej: GPIO(1,21) → pinAbs = 1*32+21=53
- Exportar el pin para su uso, escribiendo su número absoluto en /sys/class/gpio/export:

  # echo 53 > /sys/class/gpio/export
- Indicar la dirección (entrada/salida) escribiendo in ó out en /sys/class/gpio/gpioPINABS/direction:

  # echo out > /sys/class/gpio/gpio53/direction
- Escribir valor deseado (high/low) en /sys/class/gpio/gpioPINABS/value:

  # echo high > /sys/class/gpio/gpio53/value

3.2.2. Manejo de salidas PWM

Las 3 salidas analógicas en modo PWM están multiplexadas con pines de GPIO, por lo que debe tenerse cuidado de seleccionarlas adecuadamente.

Falta

3.3. Librería de E/S

Para facilitar el trabajo con los pines de E/S, se ha desarrollado una librería que permite realizar las distintas operaciones de forma sencilla.

Para utilizarla, deben descargarse de la página web los archivos "io_defs.h" y "libBeagleboneIO.a", y guardarlos en un directorio del sistema Linux (usaremos en los ejemplos el directorio /home/developer/proyectos/Qt5OnBB/Libs/BBIO.

3.3.1. Configuración de proyecto

Añadir las líneas siguientes en el archivo .pro del proyecto:

```
LIBS += -L/home/developer/proyectos/Qt5OnBB/Libs/BBIO –lBeablegoneIO
INCLUDEPATH += /home/developer/proyectos/Qt5OnBB/Libs/BBIO
```

Incluir el archivo "io_defs.h" en el código fuente en C:

```
#include <stdio.h>
#include <stdlib.h>
#include "io_defs.h"
```

3.3.1. Manejo de E/S digital

Las funciones para manejo de E/S digital son las siguientes:

`int InitDigitalIO(enumDIOpin pin,enumDIOtype io);`

Descripción:

Configura un pin de las patillas de expansión como salida digital

Parámetros:

pin = Nº de pin del conector de expansión a configurar. Indicar con PIN_Px_yy, donde x es 8 ó 9, e yy el nº de pin (2 dígitos). También se pueden utilizar los valores LED_USRz, con z de 0 a 3, sólo para salida.

io = Modo de funcionamiento: DIO_INPUT ó DIO_OUTPUT.

Valor devuelto:

Entero con identificador a utilizar en las siguientes llamadas, o -1 si hay error.

Ejemplo:

Configurar el pin 5 del conector 8 para entrada:

```
int p85;
p85=InitDigitalIO(PIN_P8_05,DIO_INPUT);
```

`int ReadDigitalInput(int pinId);`

Descripción:

Lee el valor actual de un pin de E/S configurado como entrada digital.

Parámetros:

pinId = Identificador de pin devuelto por la función InitDigitalIO().

Valor devuelto:

Valor del bit de entrada (0 ó 1), o negativo si no se ha podido leer.

Ejemplo:

Leer el valor del pin P8_5 configurado anteriormente:

```
valor=ReadDigitalInput(p85);   // Donde valor debe ser un int
```

`int WriteDigitalInput(int pinId,int value);`

Descripción:



Escribe el valor deseado en un pin de E/S configurado como salida digital.

Parámetros:

pinId = Identificador de pin devuelto por la función InitDigitalIO().

value =Valor a establecer (0 ó 1).

Valor devuelto:

Valor escrito (0 ó 1), o valor negativo si no se ha podido escribir.

Ejemplo:

Configurar el led USR1 como salida digital y escribir un 1:

```
int led1;
led1=InitDigitalIO(LED_USR1,DIO_OUTPUT,DIO_NOPULL);
WriteDigitalOutput(led1,1);
```

int EndDigitalIO(int pinId);

Descripción:

Desconfigura el pin de E/S deseado. El identificador ya no es utilizable en siguientes llamadas.

Parámetros:

pinId = Identificador de pin devuelto por la función InitDigitalIO().

Valor devuelto:

0 si correcto, negativo si incorrecto.

Ejemplo:

```
int p85;
p85=InitDigitalIO(PIN_P8_05,DIO_INPUT);
… usar p85 …
EndDigitalIO(p85);
… no usar más p85 …
```

3.3.2. Manejo de entrada analógica

Las funciones para manejo de entradas analógicas son las siguientes:

int InitAnalogInput(enumAIpin pin);

Descripción:

Habilita entrada analógica en pin del conector de expansión P9.

Parámetros:

pin = Nº de pin del conector de expansión a configurar. Indicar con PIN_P9_yy, donde yy es el nº de pin (2 dígitos). También se pueden utilizar los valores AIN0 a AIN6.

Valor devuelto:

Entero con identificador a utilizar en las siguientes llamadas, o -1 si no es posible la configuración.

Ejemplo:

Configurar el pin 37 del conector 9 (AIN2) para entrada analógica:

```
int p9_37;
p9_37=InitAnalogInput(PIN_P9_37);
```

`int ReadAnalogInput(int pinId);`

Descripción:

Lee el valor actual de un pin de entrada analógica.

Parámetros:

pinId = Identificador de pin devuelto por la función InitAnalogInput ().

Valor devuelto:

Valor 0 a 4095, o negativo si no se ha podido leer.

Ejemplo:

Leer el valor del pin P9_37 configurado anteriormente:

```
valor=ReadAnalogInput(p9_37); // Donde valor debe ser un int
```

`int EndAnalogInput(int pinId);`

Descripción:

Desconfigura el pin de entrada analógica. El identificador ya no es utilizable en siguientes llamadas.

Parámetros:

pinId = Identificador de pin devuelto por la función InitAnalogInput ().

Valor devuelto:

0 si correcto, negativo si incorrecto.

Ejemplo:

```
int p9_37;
p9_37=InitAnalogInput(PIN_P9_37);
… usar p9_37…
EndAnalogInput (p9_37);
… no usar más p9_37…
```

3.3.3. Manejo de salida PWM

Las funciones para manejo de salidas PWM son las siguientes:

`int InitPWM(int period_ns);`

Descripción:

Establece el periodo en ns para las salidas PWM. Debe ser la primera función llamada, antes de configurar otras de PWM.

Parámetros:

period_ns = Periodo en ns.

Valor devuelto:

Entero igual a 0 si no hay error, o -1 si hay error.

Ejemplo:

Configurar perido de 5 ms para salidas PWM:

```
InitPWM(5000000);
```

`int InitPWMOutput(enumPWMpin pin);`

Descripción:

Habilita salida PWM en la patilla del conector de expansión P8 ó P9.

Parámetros:



pin = Nº de pin a configurar. Indicar con PIN_Px_yy, donde x es el nº de conector (8 ó 9) e yy es el nº de pin (2 dígitos). Ver pines válidos en 3.1.3.

Valor devuelto:

Entero con identificador a utilizar en las siguientes llamadas, o -1 si hay error.

Ejemplo:

Configurar el pin 13 del conector 8 (EHRPWM2B) para salida PWM:

```
int pwm;
pwm= InitPWMOutput(PIN_P8_13);
```

---

int WritePWMOutput(int pinId,float duty);

Descripción:

Escribe el valor de duty en la salida PWM.

Parámetros:

pinId = Identificador de pin devuelto por la función InitPWMOutput().

duty = Valor de duty deseado (0-1)

Valor devuelto:

0 si correcto, o negativo si no se ha podido escribir.

Ejemplo:

Escribir un duty del 25% en el pin P8_13 configurado anteriormente:

```
WritePWMOutput (pwm,0.25F);
```

---

int EndPWMOutput (int pinId);

Descripción:

Desconfigura el pin de salida PWM. El identificador ya no es utilizable en siguientes llamadas.

Parámetros:

pinId = Identificador de pin devuelto por la función InitPWMOutput().

Valor devuelto:

0 si correcto, negativo si incorrecto.

Ejemplo:

```
int pwm;
InitPWM(5000000);
pwm= InitPWMOutput(PIN_P8_13);
… usar pwm…
EndPWMOutput (pwm);
… no usar más pwm …
```